

2do Parcial - NoSQL - Concurrencia y Recuperabilidad

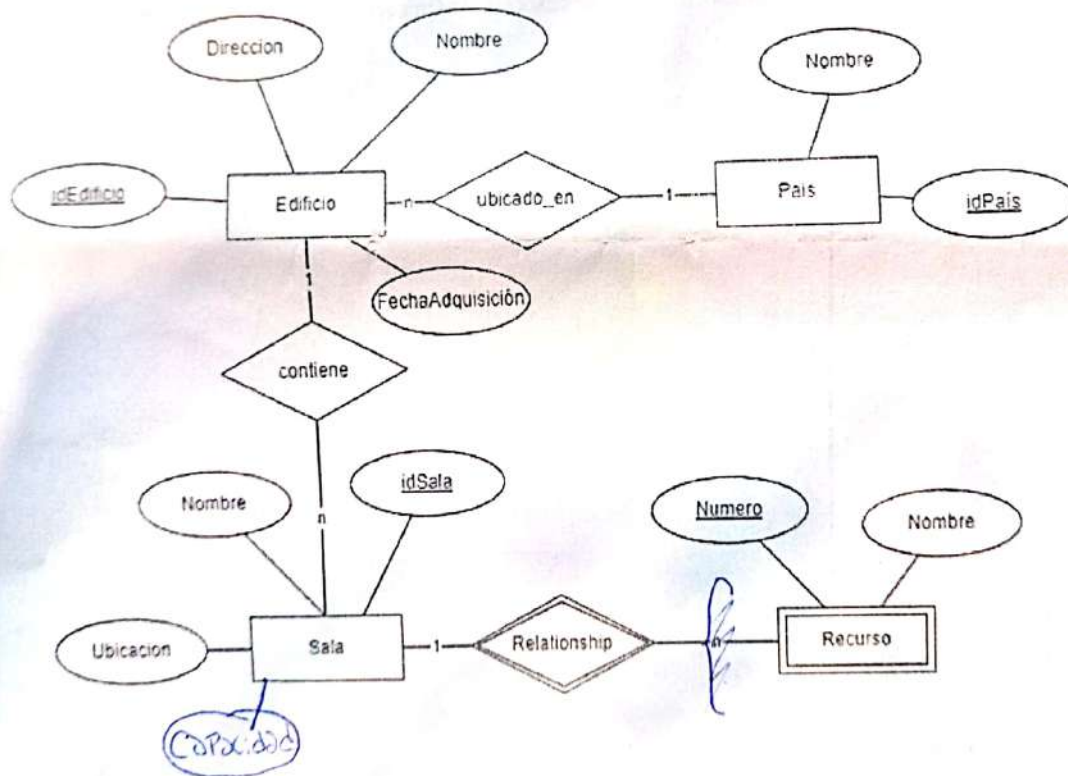
24/06/2022

Régimen de aprobación: se aprueba con 7 (siete) Ejercicio 1) 5 pts Ejercicio 2) 3 pts Ejercicio 3) 2 pts.

1) NoSQL

Dado el siguiente Diagrama de Entidad Relación que representa un modelo conceptual de parte de un sistema que administra los edificios y salas de una ONG internacional, se pide:

- Realizar el Diagrama de Interrelación de Documentos. Fundamentando cada elección (por qué se incrusta o se referencia, etc). Tomar en cuenta que cuando se consultan datos de edificios siempre se quiere saber de qué país es. Una consulta muy común es saber los datos de los edificios de un país dado.
- Realizar los diagramas de Chebotko de las siguientes consultas (justificar cada paso)
 - Nombre y Ubicación de todas las salas de edificios adquiridos en una fecha dada.
 - Las salas con una capacidad mayor a determinado número.



2) Concurrencia y Recuperabilidad

a) Dada la siguiente Historia H_1 :

$H_1 = \{w_1(A); w_2(D); r_1(B); u_1(A); r_1(A); r_1(C); r_1(B); u_3(D); u_3(A); u_3(C); w_2(C); u_2(A); u_3(B); u_1(B); w_1(B); u_2(C); c_2; w_1(E); r_1(D); u_1(E); c_1; w_1(E); w_1(C); u_3(B); u_3(E); u_3(C); u_3(D); c_4; c_5\}$

Se pide:



- i) Hacer el grafo de precedencia de H_1 , indicar si es serializable y en caso afirmativo indicar todas las historias seriales equivalentes
- ii) Clasificar H_1 con respecto a recuperabilidad: ¿Es recuperable? ¿Evita aborts en cascada? ¿Es estricta? Justificar las respuestas
- iii) La historia H_1 : ¿Es 2PL? ¿Es 2PL estricta? ¿Es 2PL riguroso? Justificar las respuestas

b) Dada la siguiente Historia $H_2 = r_2(A); r_4(B); w_2(B); r_3(A); w_4(B); c_2; w_1(A); w_3(A); c_1; c_3; r_4(B)$

Suponiendo que $TS(T1) = 400$, $TS(T2) = 200$, $TS(T3) = 300$ y $TS(T4) = 100$

y sabiendo que T1 escribe A=12, T2 escribe B=31, T3 escribe A=5, T4 escribe B= 52.

Además los timestamp de cada ítem comienzan en cero y en caso de finalizar exitosamente una transacción se realiza inmediatamente commit antes que cualquier operación de otra transacción.

- i) Indique, utilizando un planificador optimista con timestamp multiversión (MVTO), que sucede para cada acción de H_2 y que valor final quedan en B y en A.
- ii) Si T2, en lugar de commit realiza un abort, ¿qué cambia en las acciones y que valor final queda en B?

c) Se tiene el siguiente log para un planificador con Checkpoint No-Quiescente con ReDo Logging:

1	< START T1 >
2	< START T2 >
3	< T1, X, 15 >
4	< COMMIT T1 >
5	< T2, Y, 6 >
6	< START T3 >
7	< T3, S, 23 >
8	< T2, Y, 18 >
9	< START CKPT (T2,T3) >
10	< T2, Z, 31 >
11	< T3, R, 12 >
12	< START T4 >
13	< T4, T, 3 >
14	< T4, V, 8 >
15	< END CKPT >
16	< COMMIT T2 >
17	< T4, Q, 71 >
18	< COMMIT T3 >
19	< COMMIT T4 >

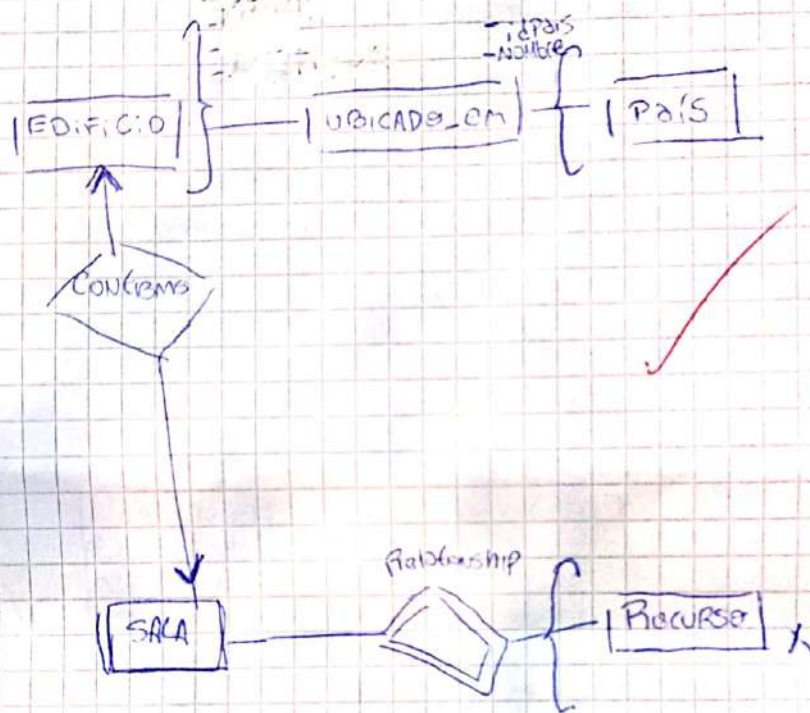
Se pide

- i) Suponga un crash luego del paso 18. Describa qué hace el recovery manager explicando cómo queda el log al finalizar y cual es el valor final de los ítems afectados por las transacciones.
- ii) Suponga un crash luego del paso 12. Describa qué hace el recovery manager explicando cómo queda el log al finalizar y cual es el valor final de los ítems afectados por las transacciones.

EJERCICIO 1:

a) Realizar el diagrama de interrelación de documentos

Se omiten los atributos en pos de la legibilidad

JUSTIFICACIONES:

INCORPORACIONES ENTRE EDIFICIO Y PAÍS: El enunciado no dice que

- 1) CUANDO se consulta un edificio se quiere saber su país. Si tan solo referenciamos al país y no incorporásemos tendríamos que hacer otro más para saber el nombre. Por otro lado, como me más interesan los datos de los edificios varados en el país restringimos a que solo me diga el nombre y el id país.
- 2) Una consulta muy común es saber los datos de los edificios de un país dado. Entiendo por este que me interesa toda la información de los edificios por lo tanto realizo una incorporación.

3) Como un recurso es una entidad debil de una sala me mas interesa crear un documento por el mismo. Ademas, me mas ~~pero~~ ^{dicen} exclusivamente que hacer embeddings por recurso sea mas frecuente como por validar la existencia de su propio documento. Por lo tanto, en este caso se decide incrustar el recurso embeddo. ✓

4) Referencias entre edificio y sala. En este caso la sala me es una entidad debil por lo cual el documento anterior me aplica. Tampoco nos dicen que dada una sala nos interesa la informacion del edificio ni viceversa, por lo tanto me hay que incrustar, lo que si hacemos es mantener la referencia entre ambos. ~~Para~~ ✓



o sea, por entidad

Justificaciones:

M2: Dado que nos estan pidiendo los datos dada una fecha dada esta debe ser marcada como K ✓

M3: No hay ningun requisito en lo se a desigualdades ✓

M4: No se pide que la informacion este ordenada. ✓

M5: Falta mapear las claves de edificio y sala. Las marca CP ✓

M6: Como me trae la ubicacion me pueden saberse a traves de fecha y quisiera no los marcar con S. ✓

NOTA

BODICHO SERAGAN

ii)

SALA SEGUN CAPACIDAD
CAPACIDAD
ID SALA
NOMBRE
UBICACION
RECURSO
RECURSO NOMBRE
RECURSO NOMBRE

USUARIO

SALA SEGUN CAPACIDAD
CAPACIDAD
ID SALA
ID EDIFICIO
RECURSO NOMBRE
RECURSO NOMBRE
UBICACION
NOMBRE

M1

II
CAPACIDAD
K

M1

II
CAPACIDAD
CA

M5

II
CAPACIDAD
ID SALA
NOMBRE
UBICACION
RECURSO, NOMBRE

M5

II
CAPACIDAD
ID SALA
ID EDIFICIO
RECURSO NOMBRE
RECURSO NOMBRE
UBICACION
NOMBRE
K
CA
CA
CA

M2 NO SE PUEDE

SALA SEGUN CAPACIDAD
CAPACIDAD
ID SALA
NOMBRE
UBICACION

M1

II
DUMMY
K

M3

II
DUMMY
CAPACIDAD
K
CA

M5

II
DUMMY
CAPACIDAD
ID SALA
NOMBRE
UBICACION
K
CA
CA

M2: COMO NO SE ESTA BUSCANDO POR SALAS DEBER CERRAR LOS CLUBS DUMMY

M3: COMO SE BUSCA POR DISPONIBILIDAD CAPACIDAD, LO MARCA COMO CA

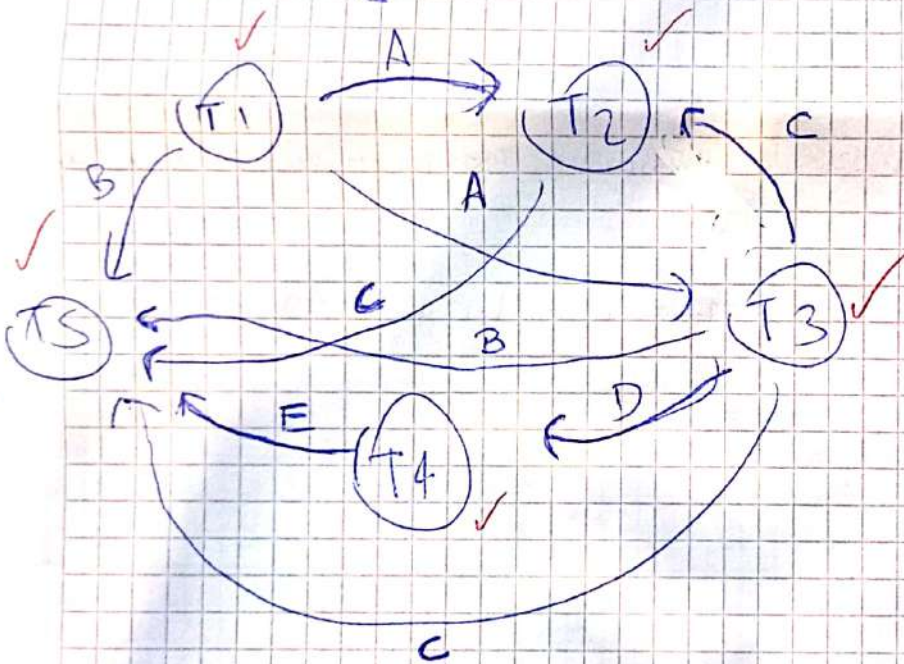
M4: NO HAY QUE

M5: Deberia falta marcar id sala

M6: No puedo saber por DUMMY los demas CAMPOS.

NOTA

a) que historia larga!!!



• T_1, T_3, T_2, T_4, T_5 ✓ @ T_1, T_3, T_4, T_2, T_5 .

ii)

- Recuperable: HACER que borre que ninguna transacción que borre otro commit antes. Voy a forzar de escribir quien los borra.
 - T_2 lee A de T_1
 - T_3 lee A de T_1
 - T_4 lee D de T_3
 - Como T_1 commita antes que T_2 y T_3 se cumple
 - Como T_3 commita antes de T_4 se cumple.
- luego, es recuperable. ✓

Abort en cascada: No, ya que T_2 lee A de T_1 antes que este haga commit ✓

No es estricto ya le mismo que antes, T_2 lee A de T_1 antes de que este haga commit ✓

iii)

2PL: ES 2PL ya que todas las transacciones no hacen un unlock antes de un lock ✓

2PL estricto: NO, Porque T_1 ~~tiene~~ libera lock de escritura sobre A antes del commit ✓

2PL riguroso NO, Porque no es estricto. ✓

[illegible]

74. Harland com Julia no dia que o único ~~entrevista~~ avuls inmediatamente después después en al de

El final Poo A y B os el del timestamp ms reciente. Para A queda el ver 12 y B el 31.

Si T_2 elabore me cuido em melhorar os recursos das leis dando transições e a que T_1 me deixa su mior de todos meados \checkmark lo que si cuido es el valor final de P ya que quedara el escrito por T_1 en vez de de T_2 .

Ejercicio 2 C

Redo lossus:

i) El Recovery Manager va a encontrar el ~~start~~ END CKPT del paso 15
 Para luego buscar el start de ese end hasta el paso 9 y luego hasta
 el start de la transacción mas vieja para saber cuales hicieron commit y cuales no.
 Luego

- DEBE Realizar REDO EN T2 ✓
- " " " " T3 ✓
- No debe " " " T1 Porque fueron antes del checkpoint ✓
- DEBE Abortar T4 Porque no commit. ✓

El los termino con el Abort de T4 y logjando a disco. ✓

valores finales de los ítems

S: 23

Y: 18

Z: 31

R: 12

✓ y $X = 15$ por T1 commitada

ii) El Recovery manager va a encontrar el ~~start~~ CKPT y va a buscar el anterior
 (end checkpoint) (que no hay). ~~Por lo tanto~~ ✓

El Ws va a:

- DEBE Realizar Redo EN T1 (Porque no encontro el end checkpoint que verifico los commits. ✓)
- DEBE HACER Abort de T2 ✓
- " " " " T3 ✓
- " " " " T4 ✓

valores ítems.

X: 15 ✓